

MBTiles

a simple specification for portable tilesets

Tom MacWright from Development Seed

Why

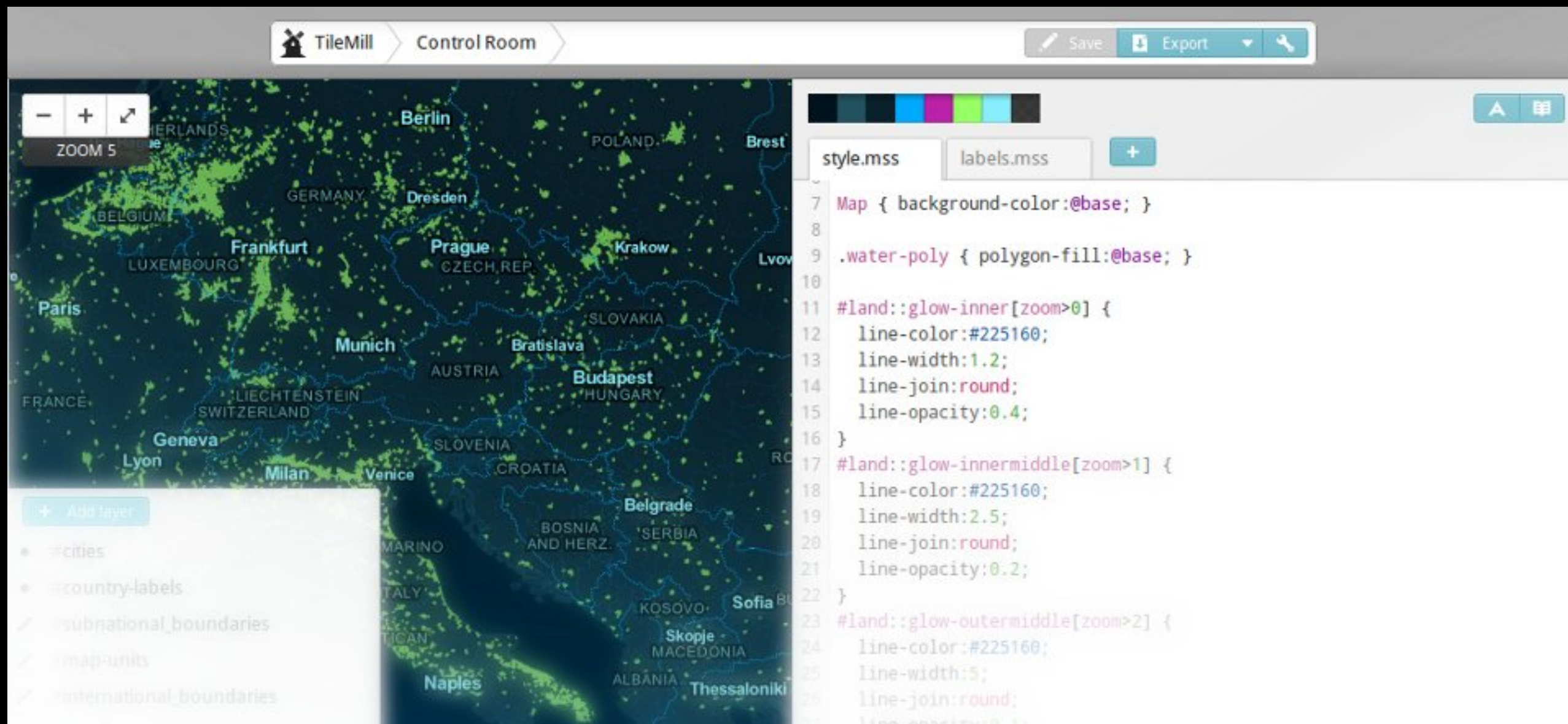
- We needed a way to store tilesets - rendered image tiles - that was easier to use and less error-prone than storing millions of files on disk
- And down the line, we needed a way for people to distribute & deploy tilesets - a standard that could work on iPads & hosted tile servers equally well

Us - MapBox & Development Seed

- We're building a fully open-source map design & publishing stack
- TileMill is a map design studio

Modest Maps-powered viewer

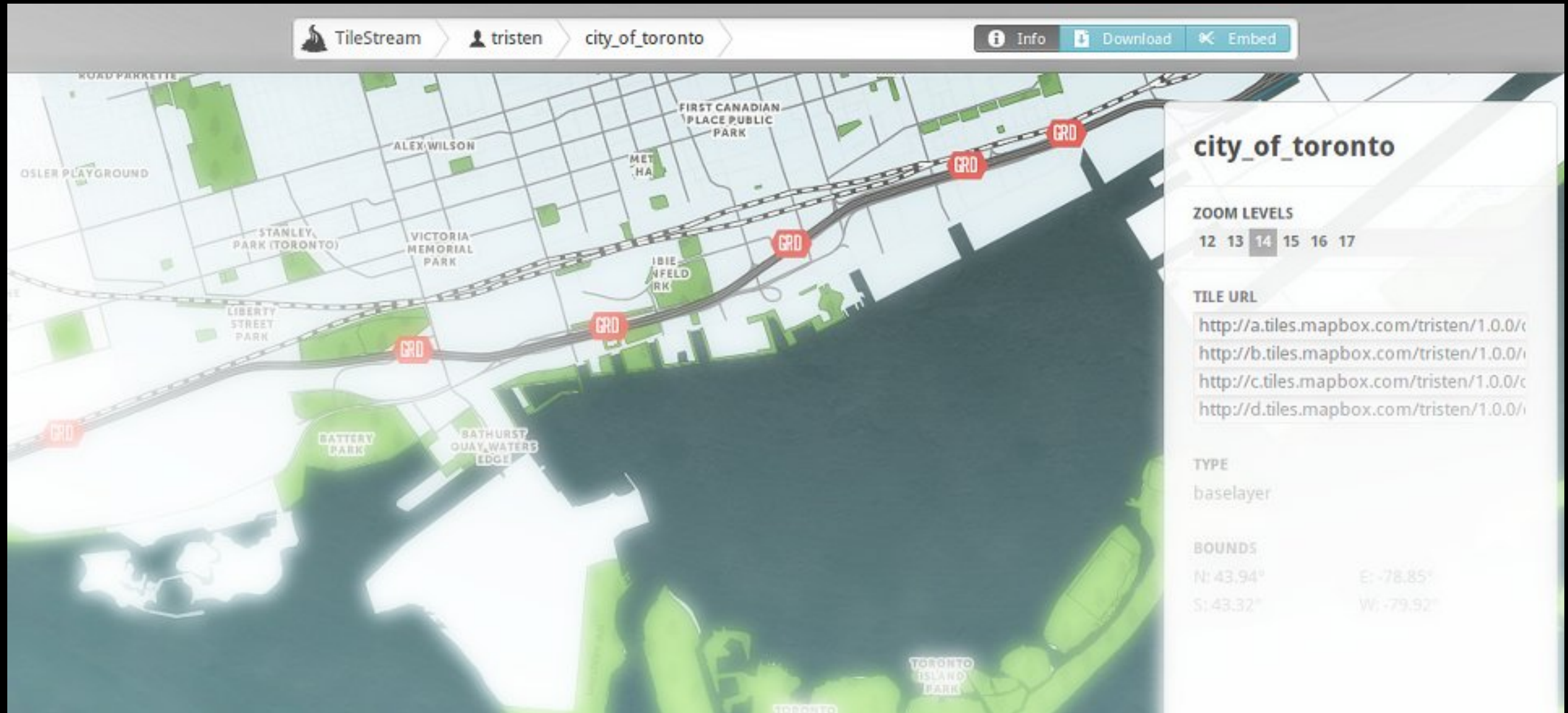
Export to **MBTiles**



TileMill

Carto - a CSS-like map language

TileStream: map hosting



MBTiles as tile storage

Ecosystem Overview

- We use **MBTiles** to store map tiles
- **Carto** is our CSS-like language for styling maps, with **Mapnik** as the targeted renderer
- **TileJSON** is our standard for describing maps. Less verbose, less XML, but similar in goal to GetCapabilities
- **Data** comes in as Shapefiles, SQLite, GeoJSON, PostGIS, KML, GeoTIFF. Not supported, with no plans: GML, GeoDatabase. We rarely see GML documents, and don't support proprietary formats.

Problems Addressed

- Filesystem inode limits prevent XYZ/TMS tiles from being stored on disk at high zoom levels
- Filesystems are often inefficient with high file/directory ratios - especially FAT32
- Transfer time is inflated - often multiplied - by per-file overhead

Specification Bounds

- MBTiles is not a tiling specification; servers can serve tiles in the Tile Map Service or XYZ schemes.
- The storage format is SQLite 3.0 and higher
- Image tiles stored as binary data, as PNG, JPG, GIF, or any other standard

The Specification

- MBTiles was announced October 8th, 2010
- Specification managed as an open-source project on GitHub beginning February 2011
- Licensed Creative-Commons.
- 1.0 specification
- 1.1 specification

Generation

TileMill	BSD	JavaScript
mbutil	BSD	Python
Arc2Earth	Proprietary	ArcScript
MapProxy	Apache	Python

Servers

TileStream	BSD	JavaScript
TileCache	BSD	Python
TileStache	BSD	Python
Brook	BSD	Ruby, PHP
MapProxy	Apache	Python

Clients

Web APIs	Modest Maps, OpenLayers, Polymaps, Leaflet, Tile5
iOS	route-me, Maptual
Android	osmdroid, Locus, Nutiteq
Windows	MBTilesCutter

Constraints

- Single-projection: Spherical Mercator, metadata in EPSG:4326
- Single-layer: multiple files represent multiple layers
- Plain database format; no extensions
- No raw data: a presentation format, orthogonal to Spatialite and others

Single-Projection

- MBTiles is geared to web mapping
- OpenLayers is *the only* popular web mapping framework that supports other projections
- We don't use or recommend OpenLayers for general consumption - it is good for complex use cases, but overkill for simple ones

Single-Projection

- Representing multiple projections is poorly implemented - we strive for auto-configuration through TileJSON
- User expectations are for interoperability, and the majority of users are on the web. Web clients don't support reprojection and base layers (Google Maps, OpenStreetMap, Bing) are *always* in Spherical Mercator

Single-Projection

- MBTiles aims to store **specific representations of data**
- Projection is an assumption, as is format (images), and method of access (tiles)

Single-Projection

- We are serving *tiles of data*
- There is very little adoption of multi-projection tile schemas in mainstream technology. OSM is the standard

Plans

- Stronger support for **presentational** vector data
- No plans to implement a 'Single File Map' in this standard - a separate standard, and multiple underlying standards would be required
- A validator for MBTiles sets could be useful

State of MBTiles

- High adoption because of simplicity
- Specification is still moving quickly
- No restrictions on usage, minimal restrictions on the spec
- Support in both open-source and closed-source (iPad apps, Arc2Earth)

Other Topics: Carto

- We developed Carto, a successor to Cascadenik, a CSS-like language for Mapnik
- Currently has a JavaScript implementation; C++ implementation incoming

Carto: Not Ready

- Carto standardization requires a formal grammar describing the language. Currently only has reference implementations.
- Cross-renderer standard will require heavy-duty reconciling of **capabilities** and **data models** which has not even started

Carto: Not Ready

- Also will require rethinking of some of the base concepts of Carto: more work is on the way to start up discussion with Cascadenik authors and authors of other style languages to find a happy medium and something that'll last

Other Topics: Single-File-Map

- Lots of discussion of a single-file-map concept
- This **isn't feasible currently** and is the **wrong problem** to be focusing on
- A single file map would be a container format of a data format, styling language, and metadata

Single-File-Map

- So, when we've standardized a styling language, data format, and metadata format, then we'll have a single-file-map with any easy-to-find container, like ZIP.
- But we **don't have any of those things.**
- We'll have a de-facto single-file-map via progress in developing TileMill: but this will take serious time to generalize